MoBIES

VANDERBILT UNIVERSITY
INSTITUTE FOR SOFTWARE INTEGRATED SYSTEMS

MODEL-BASED SYNTHESIS OF GENERATORS FOR EMBEDDED SYSTEMS

PI: Gabor Karsai, Co-PI: Akos Ledeczi
(615) 343-7471 gabor@vuse.vanderbilt.edu
Institute for Software-Integrated Systems
Vanderbilt University

F30602-00-1-0580
End date: 5/04

Approved for Public Release, Distribution Unlimited

2. Subcontractors and Collaborators

Collaborators

- CMU
  - Analysis Interchange Format for embedded systems
- Kestrel
  - Matlab input translator, HSIF definition
- Southwest Research
  - Modeling language for avionics applications
- Teknowledge
  - Rational Rose import facility to ESML
- U Michigan
  - Modeling environment for embedded systems
- U Penn & SRI & UCB OEP
  - Hybrid System Interchange Format
- OEP experiments
  - Tool integration support

3. Project objectives

Problem:

Model-based development of embedded systems requires tools for transforming models for analysis and synthesis

→ How to build reusable technology for generators (a.k.a. model interpreters) for embedded systems?

Contribution to MoBIES:

- Meta-programmable generator technology
- Tools:
  - **Meta-programmable modeling environment (GME)**
  - **Reusable translator framework (UDM)**
  - **Design tool integration framework (OTIF)**
- Examples:
  - **ESML: Modeling language for avionics OEP**
  - **ECSL: Modeling language for automotive OEP**
  - **Interfaces and translators: HSIF, AIF, ESCM**

Success criteria

- Affordable and usable tools for building model translators
- Functional OTIF

3. Project Objectives

Relevance of Model Translators:

Linkage between models and their interpretation

Use cases:

- Translation into analysis formalisms
  - See ESML -> AIF
- Synthesis of system configuration information
  - See ESML -> XMLCONFIG
- Generation of code (via instantiating code templates)
  - See ECSL -> C code
- Tool integration using semantic translators
  - See OTIF
- Custom mapping from OMG MDA’s PIM to PSM
  - Mapping might be domain- or product-line specific

Support

VANDERBILT UNIVERSITY
INSTITUTE FOR SOFTWARE INTEGRATED SYSTEMS

MoBIES Program Milestone Support Matrix

Project Name: Model-based Synthesis of Generators	Current Official Milestone Completinon Date	Does your project support this milestone? (Y or N)	Identify your specific activities that support(ed) this milestone.	Specify completion date for activities that support(ed) this milestone
Task 1: Multiple-View Modeling of Physical Constraints				
1. Demonstrate ability of modeling cross cutting physical constraints	1QFY01	Y	The meta-programmable Generic Modeling Environment (GME) supports modeling of cross-cutting physical constraints. Example: power consumption across multiple HW/SW configurations.	30-Sep-00
2. Demonstrate ability to model domain specific model semantics	4QFY01	Y	GME provides support for capturing static semantics in the form of constraints	30-Sep-01
3. Demonstrate ability to customize generic modeling tools	4QFY01	Y	GME is meta-programmable	30-Sep-01
4. Demonstrate ability of propagating constraints among views	2QFY02	Y	GME supports constraints across multiple views. Example: avionics challenge problem, integrity constraints across views	31-Dec-01
5. Demonstrate ability to integrate different models of concurrency	2QFY02	N		
6. Demonstrate ability to integrate domain specific modeling tools	2QFY02	Y	Support for tool integration activities: translators and integration infrastructure components.	31-Mar-02
7. Demonstrate ability to compose multiple view models	4QFY02	Y	GME provides support for multiple view modeling. Example: Avionics OEP component, (SW) interactions, and (HW) configurations views	31-Jan-02
8. Demonstrate ability to verify multiple-view models	4QFY03	N		
Task 2: Model-Based Generation Technology				
1. Demonstrate ability to mathematically model generators	4QFY01	Y	Modeling translator inputs and outpus, and translation activities (using graph rewriting rules)	30-Sep-01
2. Demonstrate ability to customize frameworks with generators	4QFY02	Y	Configuration Generator for the Avionics OEP Bold Stroke framework	31-Jan-02
3. Demonstrate ability to compose generators from components	2QFY02	Y	Componentatization of generators, reusable/generic components	31-Mar-02
4. Demonstrate ability to generate embedded software from models	4QFY02	Y	Code generator for a dataflow/stateflow modeling language. Example: Automotive OEP code generator	1-Mar-02
5. Demonstrate ability to synthesize generators from formal spec.	2QFY03	Y	The generator's code will be synthesized from the abstract models of generator input and output, and the graph rewriting	31-Mar-03
6. Demonstrate ability to synthesize systems from models	4QFY03	Y	The generator technology will be used to demonstrate that it can be used to build a tool that synthesizes the entire system	30-Sep-03
7. Demonstrate ability to guarantee properties of generated systems	4QFY04	Y	3rd party tools integrated into the generation process will be used to verify properties	30-Sep-04
Task 3: Framework Composition Technology				
1. Demonstrate ability to customize multiple frameworks from models	4QFY02	N		
2. Demonstrate ability to generate interface code to couple frameworks	4QFY03	Y	Our generator technology could be applicable here.	30-Sep-03

5. Tool Description

GME: Meta-programmable modeling environment

- Generic modeling framework for constructing domain-specific modeling environments
- Inputs: user input, models from other tools
- Outputs: models
- Metamodel: see GME meta-model in doc
- Interface now
  - U Michigan AIRES (Meta & API)
  - SWRI ASC (Meta & API)
  - Tek Rose Export (XML)
  - UCB Ptolemy (HSIF/XML) – UCB group
  - UPenn Charon (HSIF/XML, Model text) – UPenn group
  - CMU PIHA (HSIF/XML, Model text)
  - SRI SAL (HSIF/XML) – SRI group
- Interface in 6 mos
  - CMU TimeWiz (XML)
  - Teja (HSIF/XML)
  - SHIFT (HSIF/XML)
- NonMoBIES
  - Matlab SL/SF (Translator & XML)
  - R Rose (XMI & Translator)

5. Tool Description

UDM: Universal Data Model facility

- Meta-programmable package for building generators/translators
- Inputs: Data model in UML class diagrams
- Outputs: C++ API implementation of data model, usable with various backends
- Metamodel: see UDM meta-model in doc
- Interface now/in 6 mos, MoBIES & non-MoBIES
    - Any tool that
        1. uses XML as data representation, or
        2. for which a back-end exists/can be developed

UMT: Universal Model Translator

- Meta-programmable tool for building the rewriting engine generators/translators
- Inputs: Translation models + UDM Data models
- Outputs: Interpretive engine (now), C++ code that implements the engine (later)
- Metamodel: Possible (bootstrap!)
- Interface now/in 6 mos, MoBIES & non-MoBIES
    - Same as UDM, as the rewriting engine relies on UDM

5. Tool Description

(M)OTIF: (MoBIES) Open Tool Integration Framework

- Reusable framework components and meta-programmable generators for building tool integration solutions
- Inputs:
  - Translation models + UDM Data models
  - Hand-coded components (e.g. physical tool adaptors)
- Outputs:
  - Tool chain instance: support framework that connects the tools
- Metamodel: Yes: UDM + XLT models and model of tool chain
- Interface now/in 6 mos, MoBIES & non-MoBIES
  - Tools that have an UDM interface (HSIF, etc.)

6. OEP Support: Automotive OEP

Role:

- MATLAB import translator & prototype code generator provider
- HSIF contributor
- Tool integration solution provider
- Design space exploration tool provider

Midterm experiment:

- Import translator for Matlab SL/SF into ECSL
- Prototype code generator for ECSL
- Model compiler example (design space explorer)

Contributions to experiment planning & tech assessment:

- Tool chain definition, HSIF definition, integration technology

Tech POC:

- P. Varaiya, A. Girard, P. Griffiths (UCB)
- K.Butts, B. Milam (Ford)

6. OEP Support: Avionics OEP

Role:

- Provider of ESML modeling environment, translators, and generators
- AIF contributor
- Tool integration solution provider

Midterm experiment:

- Import translator from R Rose into ESML
- ESML modeling environment
- Generators for XMLConfig and AIF
- Analysis tool/report generator for ESML

Contributions to experiment planning & tech assessment:

- Tool chain definition, AIF definition, integration technology

Tech POC:

- D. Sharp, M. Schulte, W. Roll

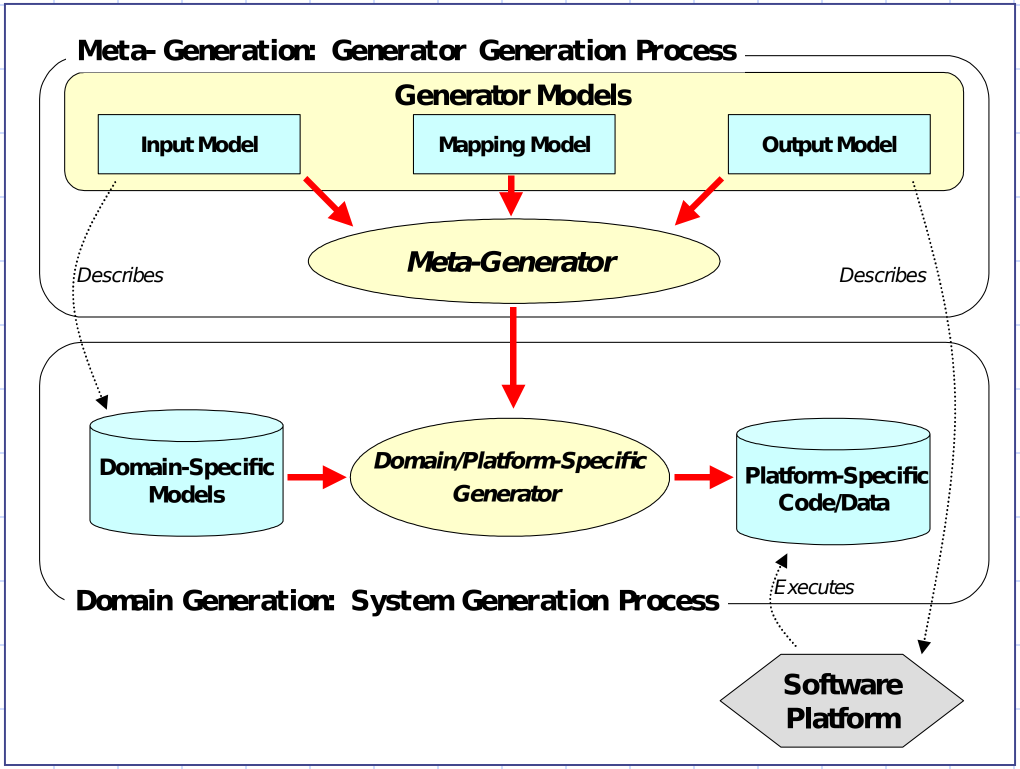
7. Project Status

Technical approach

Key concept:

Capture the semantic mapping between the two type systems/semantics in a mapping model. This mapping describes a generation process.

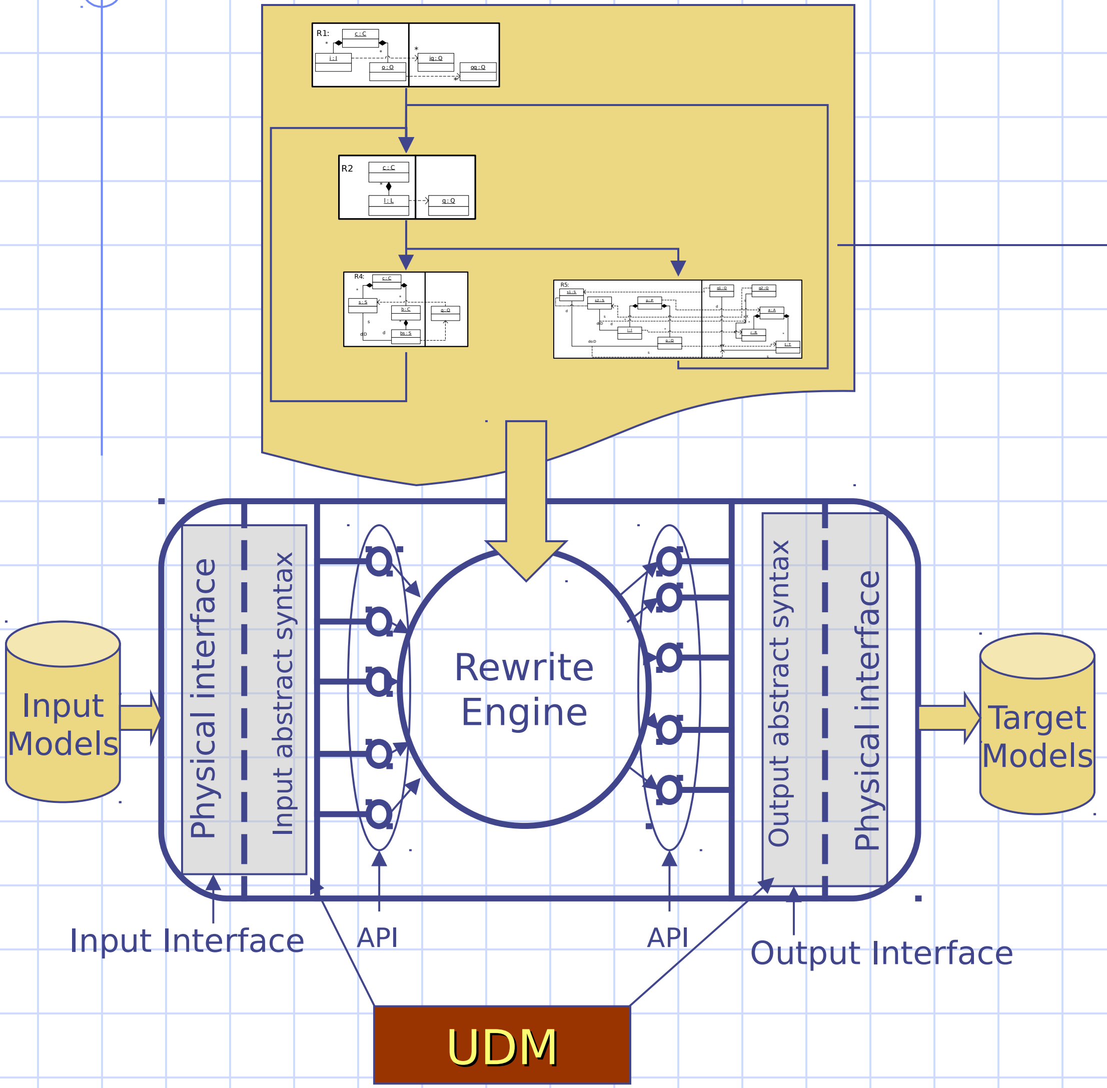
Synthesize generators from the model of the input/target/generation process



Meta- Generation: Generator Generation Process
Generator Models
Input Model
Mapping Model
Output Model
Meta-Generator
Describes
Describes
Domain-Specific Models
Domain/Platform-Specific Generator
Platform-Specific Code/Data
Executes
Domain Generation: System Generation Process
Software Platform

7. Project Status

Translation via graph rewriting



<u>1st implementation:</u>
“Interpretative” approach

<u>Translation models:</u>

1. Context-sensitive graph rewriting rules
   - [guard] Graph Pattern
     - <ANCHR>
     - =>
   - New Graph
2. Rewriting “program”: (control flow)
   - Sequence
   - Decision points: tests
   - “Pass-along” objects

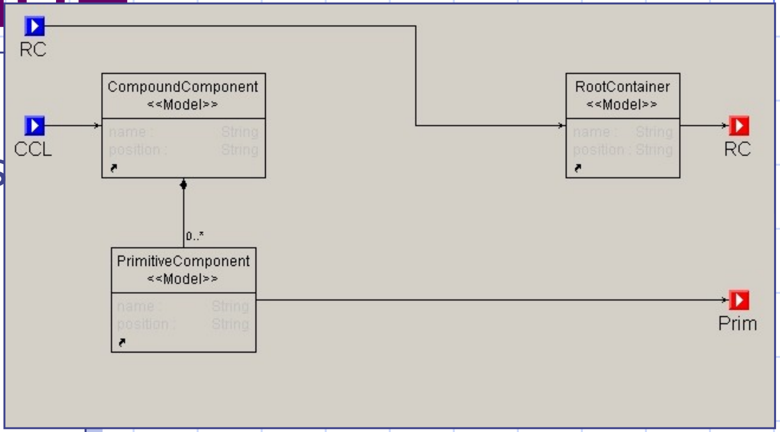
7. Project Status
Rule pattern language

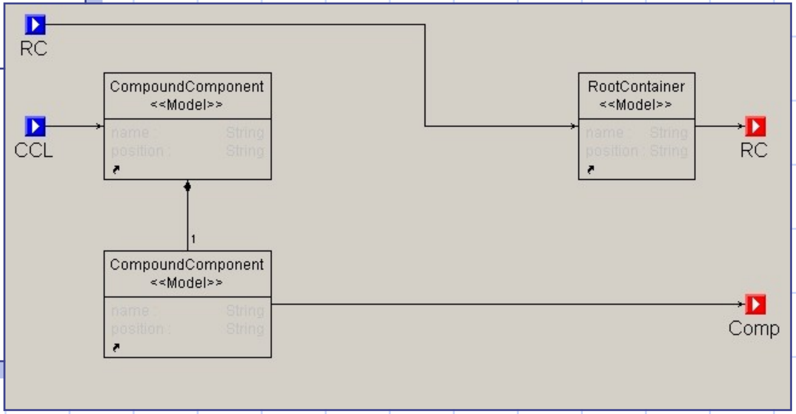
Concepts:

- Typed pattern variables -> Class
- Variable cardinality
- Typed pattern links -> Associations
- Link cardinality
- Guards -> Expressions over attributes

Algorithms:

- “Simple” matching
- Fixed cardinality matching
- Unlimited cardinality matching





7. Project Status
Rule action language

Concepts:

Typed target variables -> Classes

Typed target links -> Associations

Actions:

CreateNew: make target objec

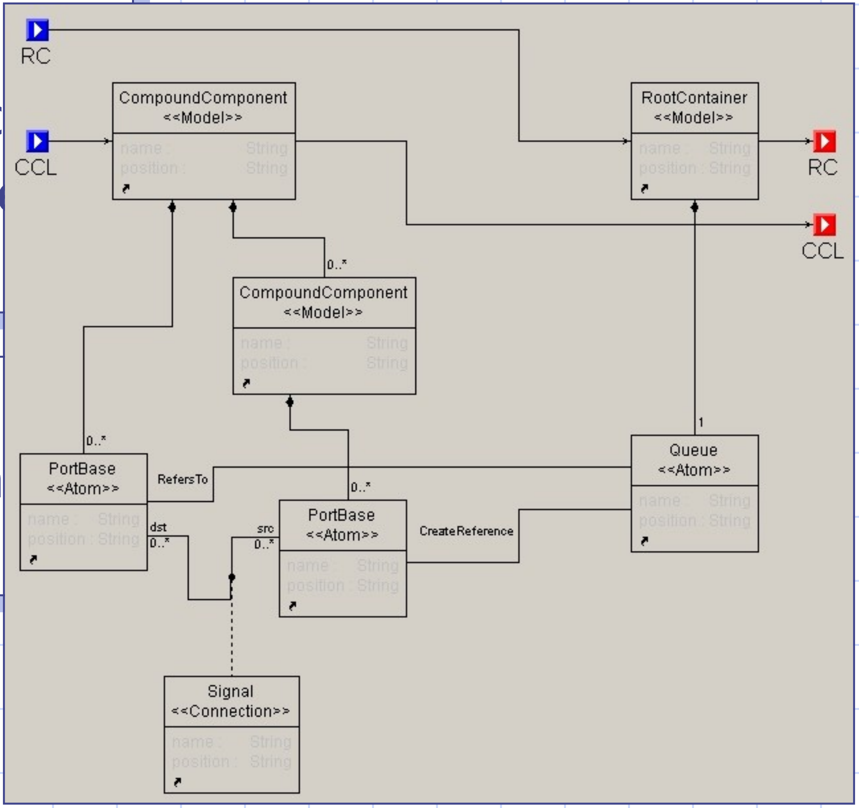
Refer: reference existing targ

ReferElseCreate:ref/create

Algorithms:

Create portion of target graph

Link it to context



7. Project Status

Rule language

Concepts:

LHS: Pattern

RHS: Portion of the target

Parameters:

- Input: pattern variables bound by previous rules
- Output: pattern variables bound in this rule, passed on

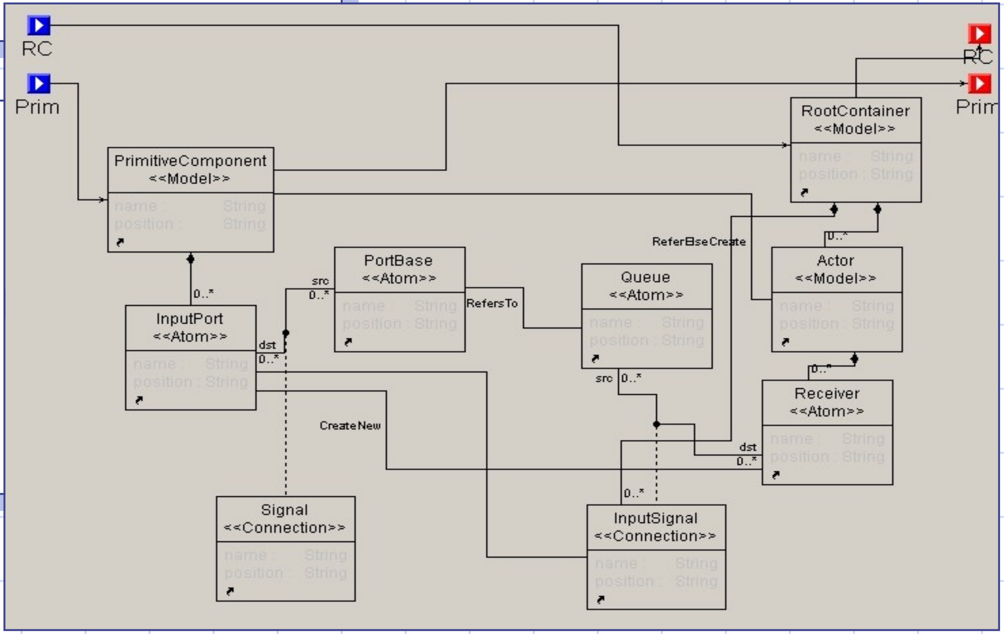
Algorithm – Rule firing

Bind LHS

Match LHS – if fail, retur

Execute actions

Bind RHS



7. Project Status

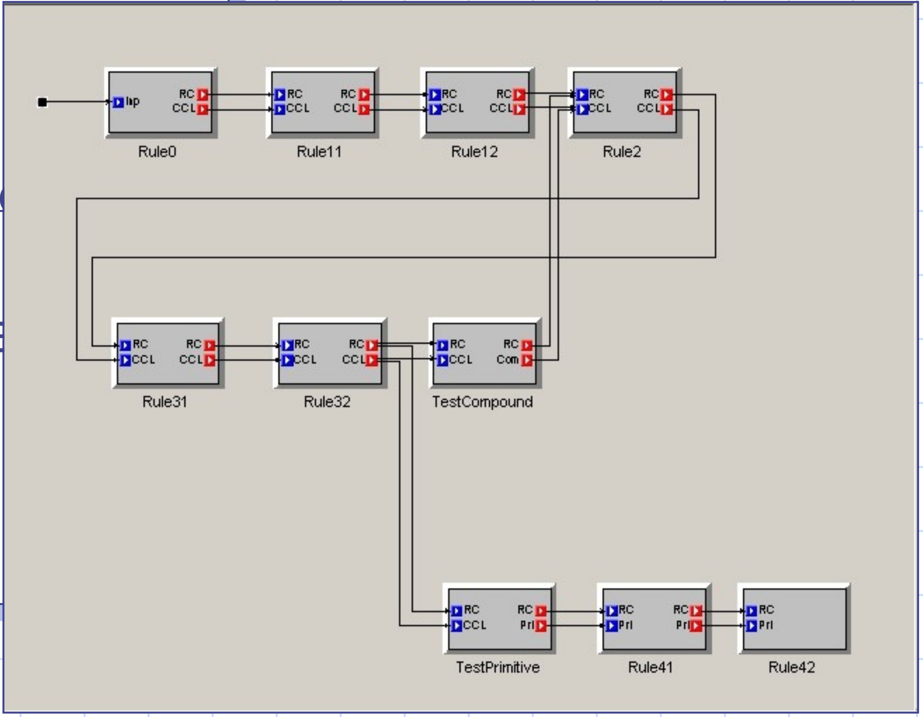
Traversal language

Concepts:

- Rules: encapsulation with in/out objects
- “Pass-along” connections
- Sequencing
- Branching – “test and proce

Algorithm – Traversal

```
while rules ready to execute
    Bind rule
    Fire rule
    Determine follow-up rules
```



7. Project Status
Core research

Status:

- Prototype is being tested and improved

Next steps:

- Refinements for the rule & traversal language
- Guards over rules
- Attribute processing
- Practical application in real translators
- Embeddable generators

Milestone:

- Base technology for mathematical modeling of translators
- Componentization of generators

Publications:

- “Model reuse with metamodel based-transformations”, ICSR-7.
- ESML: An Embedded System Modeling Language, DASC-

7. Project Status
Updates and OEP support

Core technology:

GME: constraint checker update

UDM:

- performance improvements,
- new implementation in Java and C#,
- new features: array-valued attributes
- constraint checker

Avionics OEP:

ESML revision and extensions

Upgrades to generators, translators, and analysis tool

Utility tools:

Proxy generator

SystemID generator

New format for importing UML models: ESCM

Automotive OEP:

Bug fixes and upgrades to Matlab SL/SF translators

Prototype code generator for ECSL:

Simulink discrete time blocks and Stateflow blocks into C

7. Project Status HSIF

Tools that can be used with HSIF:

Tool	Modeling	Analysis	Simulation	Generator
Simulink/ Stateflow	Yes	No	Yes	Yes
Ptolemy	Yes	No	Yes	Yes
Charon	Yes	Yes	No	No
ddt	Yes	Yes	No	No
Teja	Yes	No	Yes	Yes
Checkmate	Yes	Yes	No	No
SAL	Yes	Yes	No	No
Shift	Yes	Yes	Yes	Yes
AIRES	Yes	Yes	No	No
ECSL	Yes	No	No	Yes
HSIF/GME	Yes	No	No	No

7. Project Status

HSIF

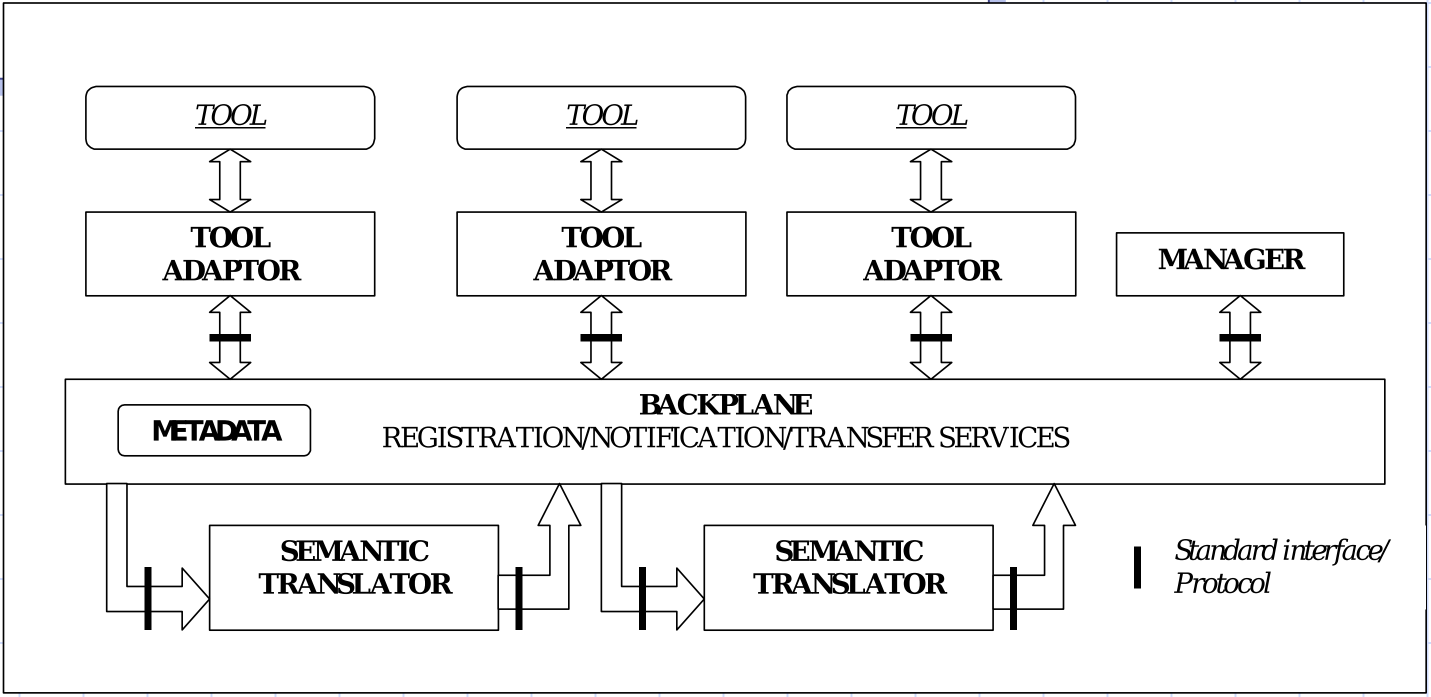
Status summary:

Tool	Export to HSIF	Import from HSIF
Simulink/Stateflow	Needed, but complex.	Probably not needed.
Ptolemy	Not planned.	Haiyang @ Berkeley is working on it.
Charon	Oleg is working on it.	Oleg is working on it.
ddt	?	?
Teja	Waiting for XML.	Waiting for XML.
Checkmate	Not planned.	Jon @ Vanderbilt has a first prototype.
SAL	?	Ashish @ SRI is working it.
Shift	?	UCB OEP group has expressed interest.
AIRES	?	?
ECSL	Needed, but complex.	Not planned.
HSIF/GME	Done.	Probably not needed.

7. Project Status

OTIF

- Architecture defined
- Protocols in OMG/CORBA
- UDM extension: CORBA as a transport layer
- Prototype: Backplane, Manager, Tool Adaptor/Lib



8. Project Plans

Next 6 months

1. Enhance Graph Rewriting Engine, use it on translators
2. Generative approach for translators
3. ESML/ECSL updates and fixes
4. OTIF enhancements
5. Prototype translators for tool integration

Performance goals

- Functional GRR engine, with 3 simple translators working:
  - 1 working, 2 designed
- Core components of the OTIF functional
  - OK
- OTIF instance created, and functional for end-to-end scenarios with 3 types of tools: modeling, analysis, and generator.
  - Modeling -> Modeling/Generator working

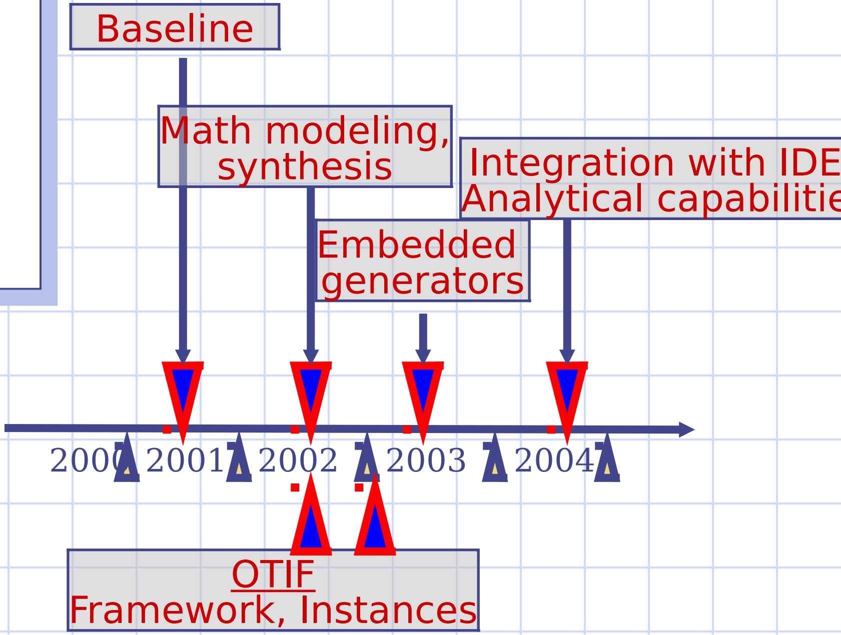
Success metrics

- Functioning translators
- Development of translators is at least 50% faster than by hand coding
- OTIF core components functional
- Tool integration solution is created at least 50% faster than by hand
- OEP success criteria satisfied

9. Project Schedule

Tasks

1. Development of model-based technology for generators: modeling and synthesis
2. Develop techniques for composable and embeddable generators
3. Develop a solution for Open Tool Integration Frameworks
4. Integration with IDEs, analysis



9. Project Schedule

Milestones in past 6 months:

- Demo ability to integrate design tools ☑
- Composition of multiple view models ☑
- Customize frameworks with gen's ☑
- Compose generators ☑
- Generate embedded SW for OEPs ☑

Milestones in next 6 months:

- Robust GRE is a tool used in building translators
- Enhanced ESML/ECSL and associated tools
- Interface definitions refined: HSIF, ESCM, AIF, etc.
- Design for embeddable generators
- End-to-end tool chain for OEP(s)

10. Technology Transfer

Vehicles:

- DoD contractors:
  - Boeing, LM, Raytheon
- Software and non-software business entities using the technology
  - Daimler-Chrysler, Ford, GM, Mathworks, …
- OMG for standardization
- Graduating students

Status

- Discussions with
  - Aerospace: Boeing, LM, and Raytheon
  - Automotive: Ford, GM, Daimler-Chrysler
- Communication with various industrial entities